UTILITY PATENT APPLICATION TRANSMITTAL

(Only for new nonprovisional applications under 37 CFR 1.53(b))

Title of Invention	"METHOD AND SYSTEM FOR CHARACTER SEQUENCE CHECKING ACCORDING TO A SELECTED LANGUAGE"
Named Inventor(s)	Benoit Jules Jurion; Chakrapani Appalabattula; and Steven Chai
Attorney Docket	13237-2425
Express Mail Label No.	EL329860004US

APPLICATION ELEMENTS

1. ☐ Fee Transmittal Form
   (Submit an original, and a duplicate for fee processing)
2. ☒ Specification, Claims, and Abstract — Total Pages **38**
3. ☒ Drawings — Total Sheets **12**
4. Oath or Declaration — Total Pages **0**
   a. ☐ Newly executed (original or copy)
   b. ☐ Copy from prior application (37 CFR 1.63(d))
   (for continuation/divisional with Box 17 completed)
   [Note Box 5 Below]
   (i) ☐ DELETION OF INVENTOR(S)
   Signed statement attached deleting inventor(s) named in the prior application, see 37 CFR 1.63(d)(2) and 1.33(b).
5. ☐ Incorporation by Reference
   (usable if Box 4b is checked)
   The entire disclosure of the prior application, from which a copy of the oath or declaration is supplied under Box 4b, is considered as being part of the disclosure of the accompanying application and is hereby incorporated by reference therein.
6. ☐ Microfiche Computer Program *(Appendix)*
7. ☐ Nucleotide and/or Amino Acid Sequence Submission *(if applicable, all necessary)*
   a. ☐ Computer Readable Copy
   b. ☐ Paper Copy (identical to computer copy)
   c. ☐ Statement verifying identity of above copies

ADDRESS TO: Assistant Commissioner for Patents
Box Patent Application
Washington, D.C. 20231

ACCOMPANYING APPLICATION PARTS

8. ☐ Assignment Papers (cover sheet & document(s))
9. ☐ 37 CFR 3.73(b) Statement *(when there is an assignee)*
   ☐ Power of Attorney by assignee
10. ☐ English Translation Document *(if applicable)*
11. ☐ Information Disclosure Statement (IDS) PTO-1449
    ☐ Copies of IDS Citations
12. ☐ Preliminary Amendment
13. ☒ Return Receipt Postcard (MPEP 503)
    (Should be specifically itemized)
14. ☐ Small Entity Statement(s)
    ☐ Statement filed in prior application
    Status still proper and desired
15. ☐ Certified Copy of Priority Document(s)
16. ☐ Other: _____

17. If a **CONTINUING APPLICATION**, *check appropriate box and supply the requisite information:*
☐ Continuation ☐ Divisional ☐ Continuation-in-part (CIP) of prior application No:

18. CORRESPONDENCE ADDRESS:

Christopher J. Leonard, Esq.
JONES & ASKEW, LLP
2400 Monarch Tower
3424 Peachtree Road, N.E.
Atlanta, Georgia 30326

By: _____ Reg. No.: 41,940
Date: June 30, 1999
Telephone: 404-949-2400
Facsimile: 404-949-2499

"METHOD AND SYSTEM FOR CHARACTER SEQUENCE CHECKING ACCORDING TO A SELECTED LANGUAGE"

RELATED APPLICATIONS

This application is related to the following application, which is filed on the same day as the present application and is assigned to the same assignee as the present application:

"Method And System For Automatic Type And Replace Of Characters In A Sequence Of Characters" - serial no. _____.

TECHNICAL FIELD

This invention relates to validation of and correction of sequences of input characters according to the syntactical rules of a selected language. More particularly, this invention relates to determining whether a typed sequence of characters is a valid sequence according to the character sequence and syntactical rules of the selected language.

BACKGROUND OF THE INVENTION

In recent years, use of computers has increased dramatically worldwide. Users of computers utilize computer programs for a variety of purposes including word processing, database management, desktop publishing, and the like. Computer users are accustomed to using "checking" program modules (e.g., spell checkers and grammar checkers) that alert the user to words or sequences of characters found in the document that are questionable based on some predefined set of rules.

The written form of some languages includes sequences of complex characters and/or symbols. For example, South Asian languages like Thai, Vietnamese, and Hindi use combinations of various characters (also called simple characters herein) such as vowels, consonants, diacritics, tone marks, and accents to form complex characters. Those languages follow stringent syntactical rules that dictate which simple character is allowed next to or above or below another simple character in the composition of more complex characters used in the formation of words. In this context, a word can be composed of (a) one or more simple characters (e.g., consonant); (b) one or more complex characters (e.g., a complex character being formed by more than one simple character like a consonant and a tone mark); and (c) a combination of simple and complex characters. That is, the correct position of these simple characters in a complex character is necessary both syntactically and orthographically based on syntactical rules for each language. For example, in the Thai language, a leading Thai vowel must be followed by a consonant or a trailing vowel (also called following vowels) to form a valid Thai character. If other than a consonant or trailing vowel is input after the input of a leading vowel, then the sequence of the leading vowel and the subsequent non-consonant or non-trailing vowel character is an incorrect sequence and does not form a correct Thai language complex character or word. Similar character sequence and syntactical rules apply for other languages mentioned above, such as Vietnamese and Hindi. A problem arises for the person typing one of those languages because if the sequence of characters that should form a complex character is invalid, the complex character will

not be rendered on the screen correctly, and therefore, the complex character will be meaningless.

Techniques to verify the validity of a sequence of input characters have been implemented, but those techniques are mainly oriented towards getting a proper display of the complex character and do not address the issue of enforcing a correct input sequence of characters according to the syntactical rules defined by the selected language. In some prior art systems, the validity of the sequence of characters is determined by comparing the typed sequence with a known valid displayable sequence (i.e., a sequence of characters that are valid in accordance with the rules of a selected language). Some prior art techniques allow the display of sequences of characters that are orthographically incorrect or simply display a symbol, such as a black box, whenever the sequence was not displayable due to errors in the sequence of input characters. Also, prior art sequence checking techniques do not allow for determining accurately the sequence context of a previously input sequence of characters once the user moves the cursor to a new location in the text.

Accordingly, there is a need in the art for an efficient method and system for checking the validity of a sequence of input characters according to the syntactical rules of a selected language. There is further a need for a method and system for determining the sequence validity context of a sequence of previously typed simple characters.

SUMMARY OF THE INVENTION

The present invention satisfies the above-described needs by providing a method and system for checking the validity of a sequence of input characters according to the syntactical rules of a selected language. Each simple character is checked to determine whether that simple character may form a valid sequence of simple characters according to the

syntactical rules for the selected language to which the simple characters belong. If an input character may not be appended to the previously input sequence according to the rules of the selected language, the newly input character may be prohibited from being appended to the sequence and displayed on the user's computer. The present invention provides for editing of previously input character sequences by determining the validity of the context of sequences of characters.

Generally described, when a user types simple characters in the formation of complex characters in selected languages, such as Thai, Hindi and Vietnamese, a determination is made as to whether each newly typed simple character may be appended to the sequence of characters already typed by the user. If appending the new character to the sequence violates the syntactical rules of the selected language, the new character is prohibited from being appended to the sequence or from being displayed. If the user moves the cursor from one location to another in a document, a reconstruction of the sequence context is accomplished so that a determination can be made as to the validity of the next input character or any newly typed character.

More particularly described, one aspect of the present invention provides a method of checking a sequence of input characters according to rules of a selected language. The method includes the steps of receiving a first character, and determining whether the first character may begin a valid sequence of characters according to rules associated with the selected language. If the first character may begin a valid sequence of characters according to rules associated with the selected language, accepting the first character for display. If the first character may not begin a valid sequence of characters according to rules associated with the selected language, prohibiting accepting the first character for display.

If a second character is received, a determination is made as to whether the second character may be appended to the first character according to rules associated with the selected language. If the second

character may be appended to the first character according to the rules associated with the selected language, then the second character is appended to the first character. If the second character may not be appended to the first character according to the rules associated with the selected language, then addition of the second character to the first character is prohibited.

The step of determining whether the second character may be appended to the first character according to rules associated with the selected language includes utilizing a state transition table and assigning a first state to the first character according to the rules associated with the selected language. A second state is assigned to the second character according to the rules associated with the selected language. A determination is made as to whether the state transition table includes a state transition from the first state to the second state. If the state transition table includes a state transition from the first state to the second state, determining the second character may be appended to the first character according to the rules associated with the selected language. If the state transition table does not include a state transition from the first state to the second state, determining the second character may not be appended to the first character according to the rules associated with the selected language.

If appending the second character to the first character creates a complete sequence of characters according to the rules associated with the selected language, determining whether a third input character may begin a second valid sequence of characters according to rules associated with the selected language. If the third character may begin a second valid sequence of characters according to rules associated with the selected language, accepting the third character for display. If the third character may not begin a second valid sequence of characters according to rules associated with the selected language, prohibiting accepting the third character for display.

In another aspect of the present invention, a system is provided for checking a sequence of input characters according to rules of a selected

language. The system includes a computer program module operative to receive a first character and to determine whether the first character may be the first character of a sequence of characters according to the rules associated with the selected language. The program module is also operative to receive a second character, to determine whether the second character may be appended sequentially to the first character according to rules associated with the selected language, and to append the second character sequentially to the first character if the second character may be appended to the first character according to the rules associated with the selected language. Additionally, the program module is operative to prohibit appending the second character to the first character if the second character may not be appended to the first character according to the rules associated with the selected language.

In another aspect of the present invention, a method is provided for establishing the sequence validation context of a sequence of characters. The method includes determining the maximum number of characters that may comprise a valid sequence of characters according to the rules of a selected language. Beginning with the last simple character of a sequence of characters, a determination is made as to whether the last character is valid as a complete sequence of characters comprising a complex character. If the last character of the sequence of characters is valid as a complete sequence of characters comprising a complex character, then the context of the input character is returned as a context for a complex character. If the input character of a sequence of characters is not valid as a complete sequence of characters comprising a complex character, then a determination is made as to whether a combination of the last character and the character input immediately to the left of the last character is valid as a complete sequence of characters comprising a complex character. If the combination of the last character and the character input immediately to the left of the last character is valid as a complete sequence of characters comprising a complex character, then a context for the combination is returned as the context for a complex character.

If the combination is not valid as a complete sequence of characters comprising the complex character, then a determination is made as to whether the combination combined with the next character to the left of the combination is valid as a complete sequence of characters comprising a complex character. If not, subsequent combinations of characters are created by adding one at a time additional characters input to the left of the last subsequent combination until the maximum number of characters that may comprise a valid sequence have been combined to form a sequence of characters that may be checked for validity as a complete sequence of characters comprising a complex character. If any of the subsequent combination of characters are valid as a complete sequence of characters comprising a complex character according to the rules of the selected language, then a context for that combination is returned as the context for a complex character.

Other objects, features, and advantages of the present invention will become apparent upon reading the following specification, when taken in conjunction with the drawings and the appended claims.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a block diagram of a personal computer that provides an exemplary operating environment for an embodiment of the present invention.

Fig. 2 is a block diagram illustrating the interaction between exemplary program modules of an embodiment of the present invention with an exemplary computer application program.

Fig. 3 depicts an illustrative computer program screen shot according to an exemplary embodiment of the present invention.

Fig. 4 is a state transition table illustrating allowable sequences of simple characters in the Thai language and includes allowable character replacement sequences.

Fig. 4A is a state transition diagram illustrating allowable sequences of simple characters in the Thai language.

Fig. 5 is a state transition table illustrating allowable sequences of simple characters in the Hindi language and includes allowable character replacement sequences.

Fig. 5A is a state transition diagram illustrating allowable sequences of simple characters in the Hindi language.

Fig. 6 is a state transition table illustrating allowable sequences of simple characters in the Vietnamese language and includes allowable character replacement sequences.

Fig. 6A is a state transition diagram and chart illustrating allowable sequences of simple characters in the Vietnamese language.

Fig. 7 is a flow diagram illustrating the preferred steps for character sequence checking and type and replace for a newly input character.

Fig. 8 is a flow diagram illustrating the preferred steps for reconstructing the context of a previously input sequence of characters.

Fig. 9 is a table of character type and replace actions.

Fig. 9A is a table of character type and replace actions for the Thai language only.

Fig. 10 is a table illustrating a valid sequence of Hindi (Devanagari script) characters input on a user's keyboard and resulting complex character output.

Fig. 11 is a table illustrating an invalid sequence of Hindi (Devanagari script) characters input on a user's keyboard and resulting

complex character output where the last input character is invalid and not displayed.

Fig. 12 is a table illustrating replacement of a previously validated character input by a subsequent character input to form a invalid sequence of Hindi (Devanagari script) characters input on a user's keyboard and resulting complex character output.

DETAILED DESCRIPTION OF THE DISCLOSED EMBODIMENTS

The present invention is directed to checking whether a simple character initially may be input or may be appended to a previously input simple character or sequence of simple characters to form a sequence of simple characters that is valid according to the syntactical rules of the language to which the characters belong. As should be understood, the functionality of the present invention may be utilized with a variety of computer software application programs with which text is entered, including word processors, database programs, graphics programs, etc.

As discussed above, many languages including, for example, South Asian languages like Thai, Vietnamese, and Hindi use combinations of various simple characters such as vowels, consonants, diacritics, tone marks, and accents to form complex characters. Those languages follow stringent syntactical rules that dictate which simple character is allowed next to or above or below another simple character in the composition of more complex characters and also words. For the sake of simplicity, for the remainder of this document, the syntactical rules for a given language which are followed to determine the validity of a single character or sequence of characters will be referred to as "rules." It should be understood that the rules utilized by the embodiments of the present invention may include any set of syntactical rules required by any language for representing sounds or words by single characters or sequences

(including horizontal or vertical) of characters or symbols. Reference to specific languages and specific rules therefor is not intended to limit the scope of this document and the appended claims.

If a user edits characters that were previously input, the sequence validity context of a previously input sequence of simple characters may need to be rebuilt. Once the user places the cursor at a particular location, the functionality of the present invention looks backward a set number of characters to validate the past sequence of characters. Accordingly, the functionality then applies sequence checking to the next character input by the user based on the last character in the validated sequence. That is, by placing the cursor of the user's input program application (e.g., word processor) to the right of a previously input sequence of characters, the validity of that sequence is checked in order to allow the user to type another character which will be checked to determine whether it may be appended to the previous sequence in accordance with the rules of the selected language.

In another embodiment of the present invention, functionality is provided which attempts to replace or combine an existing simple character in a previously validated sequence with the new character being input by the user. This character type and replace functionality is referred to herein as character replacement for the sake of simplicity. For example, say the user has previously typed four correct simple characters in the process of typing a complex character comprised of six characters. Then, say the user types a fifth simple character which is incorrect. That is, the fifth character cannot combine with the previous four characters without violating the character sequence and syntactical rules of the selected language. The type and replace functionality of the present invention will try to replace and/or combine an existing simple character in the previously validated sequence (*i.e.*, one of the previously four typed characters) to allow the user to successfully type the fifth character. That is , if the fifth character can not be used to combine with the existing sequence of characters to create a new

complex character, the new character is discarded and is not displayed, and the existing complex character remains unaffected.

If the type and replace functionality succeeds, the new validated sequence of characters, including the last character typed by the user, are displayed on the user's computer screen. If the type and replace functionality does not succeed in replacing or combining a previous character with the newly typed character, the sequence is not changed, and the fifth character typed by the user is not displayed. The user must then attempt a different character as the fifth character. Accordingly, the type and replace functionality allows the user to automatically replace parts of an existing complex character or already validated sequence of simple characters without having to move the editing cursor within the complex character.

Exemplary Operating Environment

The processes and operations performed by the computer include the manipulation of signals by a processor or remote server and the maintenance of these signals within data structures resident in one or more of the local or remote memory storage devices. Such data structures impose a physical organization upon the collection of data stored within a memory storage device and represent specific electrical or magnetic elements. These symbolic representations are the means used by those skilled in the art of computer programming and computer construction to most effectively convey teachings and discoveries to others skilled in the art.

For the purposes of this discussion, a process is generally conceived to be a sequence of computer-executed steps leading to a desired result. These steps generally require physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical, magnetic, or optical signals capable of being stored, transferred, combined, compared, or otherwise manipulated. It is conventional for

those skilled in the art to refer to these signals as bits, bytes, words, values, elements, symbols, characters, terms, numbers, points, records, objects, images, files or the like. It should be kept in mind, however, that these and similar terms should be associated with appropriate physical quantities for computer operations, and that these terms are merely conventional labels applied to physical quantities that exist within and during operation of the computer.

It should also be understood that manipulations within the computer are often referred to in terms such as adding, calling, comparing, receiving, sending, reading, transferring, determining, routing, selecting, transmitting, etc. which are often associated with manual operations performed by a human operator. The operations described herein are machine operations performed in conjunction with various input provided by a human operator or user that interacts with the computer.

Referring now to the drawings, in which like numerals represent like elements throughout the several figures, aspects of the present invention and the preferred operating environment will be described. Fig. 1 and the following discussion are intended to provide a brief, general description of the suitable computing environment in which the invention may be implemented. While the invention will be described in the general context of an application program that runs on an operating system in conjunction with a personal computer, those skilled in the art will recognize that the invention may also be implemented in combination with other program modules. Generally, program modules include routines, programs, components, data structures (such as tables or tree structures), etc., that perform particular tasks or implement particular abstract data types. Moreover, those skilled in the art will appreciate that the invention may be practiced with other computer system configurations, including hand-held devices, multiprocessor systems, microprocessor-based or programmable consumer electronics, minicomputers, mainframe computers, and the like. The invention may also be practiced in distributed

computing environments where tasks are performed by remote processing devices that are linked through a communication network. In a distributed computing environment, program modules may be located in both local and remote memory storage devices.

Referring now to Fig. 1, an exemplary system for implementing the present invention includes a conventional personal computer 20, including a processor 21, a system memory 22, and a system bus 23 that couples the system memory 22 to the processor 21. The system memory 22 includes read only memory (ROM) 24 and random access memory (RAM) 25. A basic input/output system 26 (BIOS) is stored in ROM 24. The BIOS 26 essentially contains the basic routines that help to transfer information between elements within the personal computer 20 during certain computer operations, such as during start-up. The personal computer 20 further includes a hard disk drive 27, a magnetic disk drive 28 (e.g., that reads from and writes to a removable disk 29), and an optical disk drive 30 (e.g., that reads from an optical disk 31 or reads from or writes to other optical media). The hard disk drive 27, magnetic disk drive 28, and optical disk drive 30 are connected to the system bus 23 by a hard disk drive interface 32, a magnetic disk drive interface 33, and an optical drive interface 34, respectively.

The drives and their associated computer-readable media provide non-volatile storage for the personal computer 20. Although the description of computer-readable media above includes the hard disk drive 27, a removable magnetic disk 29 and a removable optical disk 31, such as a CD-ROM disk or DVD, it should be appreciated by those skilled in the art that other types of removable media which are readable by a computer, such as magnetic cassettes, flash memory cards, digital tapes, Bernoulli cartridges, and the like, may also be used in the exemplary operating environment.

A number of program modules may be stored in the drives and RAM 25, including an operating system 35, one or more program modules

36 (such as a word processing program module), other program modules 37. The operating system 35, in conjunction with the BIOS 26 and associated device drivers, provides the basic interface between the computer's hardware and software resources, the user, and program modules. A user may enter commands and information into the personal computer 20 through a keyboard 40 and an input or pointing device, such as a mouse 42.

Other input devices (not shown) may include a microphone, a key pad, a touch sensor, a joystick, a game pad, a satellite dish, a scanner, or the like. The keyboard 40, mouse 42, and other input devices are often connected to the processor 21 through a serial port interface 46, such as a game port or a universal serial bus (USB). A monitor 47 or other type of display device is also connected to the system bus 23 via an interface, such as a video adapter 48. In addition to the monitor 47, personal computers typically include other peripheral output devices (not shown), such as speakers and printers.

The personal computer 20 may operate in a networked environment using logical connections to one or more remote programmable devices, such as a remote computer 49. The remote computer 49 may be a server, a router, a peer device, or other common network node. Typically, the remote computer 49 includes many or all of the elements described relative to the personal computer 20, although only a memory storage device 50 has been illustrated in Fig. 1. The logical connections depicted in Fig. 1 include a local area network (LAN) 51 and a wide area network (WAN) 52. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets and the global Internet.

When used in a LAN networking environment, the personal computer 20 is connected to the local network 51 through a network interface 53. When used in a WAN networking environment, the personal computer 20 typically includes a modem 54 or other means for establishing

communications over the wide area network 52, such as the global Internet. The modem 54, which may be internal or external, is connected to the system bus via the serial port interface 46. In a networked environment, program modules depicted relative to the personal computer 20, or portions thereof, may be stored in the remote memory storage device 50. It will be appreciated that the network connections shown are exemplary and other means of establishing a communications link between the personal computer 20 and the remote computer 49 may be used.

Implementation of Exemplary Embodiments of the Present Invention

The present invention may be implemented as an independent character sequence checking and character replacement program or as a component or feature that is employed in conjunction with a separate computer program such as a word processing program capable of handling languages using complex characters. In an exemplary embodiment, the character sequence checking and character replacement program is embodied as a component in a dynamic-link library (DLL) that is used in conjunction with the "MICROSOFT WORD" word processing application program, which is published by the Microsoft Corporation of Redmond, Washington. It should be understood that the character sequence checking and character replacement functionality of an exemplary embodiment of the present invention may be used in conjunction with any computer program application that allows for typing or inputting language text. For example, the character sequence checking and character replacement functionality also may be used in conjunction with "MICROSOFT EXCEL," MICROSOFT POWERPOINT," "MICROSOFT OUTLOOK,"MICROSOFT ACCESS," AND "MICROSOFT OFFICE" products, all published by the Microsoft Corporation of Redmond, Washington.

Fig. 2 is a block diagram illustrating the interaction between an exemplary program module of an embodiment of the present invention with an exemplary computer application program, such as a word processing

application program. As illustrated in Fig. 2, the sequence checking and character replacement program module 210 may be called by the computer application program (e.g., word processing program) 205 for use of sequence checking and character replacement functionality. In an exemplary embodiment, the sequence checking and character replacement program module 210 may include a sequence checking feature 215 and a character type and replace feature 220. A sequence context reconstruction feature 225 is illustrated that may be utilized by the sequence checking feature 215 during text editing actions.

In an exemplary embodiment, it may be useful for any application program 205 that needs to use the functionality of the sequence checking and character replacement program module 210 to provide an operation system wide registry entry for the sequence checking and character replacement program module 210. This entry would point to the sequence checking and character replacement program module 210 for each language that requires sequence checking. Alternatively individual applications, such as application program 205 may register the sequence checking and character replacement program module 210 as a shared feature. Alternatively, the sequence checking and character replacement program module 210 may be embodied directly in the word processing application program 205.

Before describing the particular implementation of the present invention, it is advantageous at this point to describe a set of exemplary user interfaces encountered by a user of a given computer application program employing an exemplary embodiment of the present invention. Fig. 3 depicts an illustrative computer program screen shot according to an exemplary embodiment of the present invention. As shown in Fig. 3., sequence checking options are exposed to the user for optional use. The sequence checking feature 215 and the character type and replace feature 220 are optional to the user and may be selected as desired by clicking on the box to the left of the options. If desired, these features could be set to

operate by default rather than by option. Alternatively, these features may be exposed in the user's computer operating system registry only without a need for such option in this dialog. In this case, there should be a registry entry setting for each language that requires these features.

Referring still to Fig. 3, "Use Sequence checking" button 310 turns on/off the sequence checking feature 215. When this feature is turned off the user can type any character (tone, diacritic, vowel, and consonant, etc.) in any order and combination without intervention, unless some other type of grammar or spelling checking feature is separately employed. If the sequence checking feature 215 is turned off, other editing features like delete, backspace and cursor movements will behave just as they do in normal character input and editing operations such as with word processing using English text.

Selection of the "Type and Replace" button 320, turns on/off the character type and replace feature 220. In an exemplary embodiment, when the "Use Sequence Checking" 310 is turned off, the option "Type and Replace" 320 is not selectable and is grayed out. It only is selectable when "Use Sequence Checking" 310 is turned on.

In implementation of an exemplary embodiment of the present invention, the sequence checking feature 215 assists the user in entering a displayable sequence of characters according to the rules of the selected language by validating and memorizing each character entered by the user. If the character entered does not belong to any of the defined rules, it is not combined with the validated sequence of characters and it is not displayed. If the character is valid, then it is combined with the validated sequence and the new combination is displayed. Fig. 10 illustrates a valid sequence of simple Hindi characters input on the keyboard of a user's computer and the resulting output. The beginning of the sequence starts on the top line in the keyboard input column and ends with the character in the last line. In the

example illustrated in Fig. 10, each new simple character input by the user correctly may be appended to the previous sequence of simple characters. Fig. 11 illustrates an invalid sequence of simple Hindi characters. As illustrated in Fig. 11, the last character input by the user may not be appended to the sequence of simple characters previously input, and so the last character is not displayed in the output.

It should be understood that the sequence checking feature 215 operates separately from a spelling checker program for the selected language. That is, the sequence checking feature 215 assists the user in the formation of complex characters from simple characters. The user may still misspell a word in the selected language by incorrectly stringing together complex characters which are individually valid in terms of the sequence of simple characters comprising each complex character. Similarly, the sequence checking feature 215 operates separately from a grammar checker program for the selected language. That is, the user may string together complex characters or words that are individually valid in terms of the sequence of simple characters comprising each complex character or word, but that violate the grammatical rules of the selected language.

For example, say the user is attempting to type the word "computer" in the Thai language, and for the sake of example only, say the Thai language requires a separate complex character that is the equivalent to each letter making up the English language spelling of the word "computer." According to this example, the user would type a series of simple characters to form a Thai complex character that is the equivalent of the letter "c." The user would then type a series of complex characters to form a Thai complex character that is the equivalent of the letter "o."

The functionality of the present invention will determine that the sequences of simple characters typed by the user to form the complex characters representing the "c" and the "o" are valid sequences. If the user had typed a sequence of simple characters to form the Thai equivalent of the

letter "u" instead of the letter "o," (that is, the user is incorrectly spelling the word "computer" with a "u" instead of an "o") the functionality of the present invention would only insure the validity of the sequence of characters typed by the user for formation of the complex character representing the letter "u." Whether the word "computer" is spelled correctly will be determined by the user's spelling checker. Likewise, whether the word "computer" and a subsequent word typed by the user may be placed in sequence in accordance with the grammatical rules of the selected language will be determined by the user's grammar checker. The functionality of the present invention will only assist the user in correctly typing sequences of characters in the formation of individual complex characters and not words.

The sequence checking feature 215 of the present invention is implemented as a state transition machine, which is based on a table of transitions. Prior to typing text into a word processing document, for example, a specific language, such as Thai language, is selected by the user. This language will specify to the sequence checking feature 215 which table of transition to use. The state transition table implemented by the sequence checking feature 215 will provide the sequence checking feature 215 with allowable transitions from character to character based on the rules for the selected language.

Fig. 4 is a state transition table illustrating allowable sequences of simple characters in the Thai language and includes allowable character replacement sequences. Fig. 4A is a state transition diagram illustrating allowable sequences of simple characters in the Thai language. Fig. 4A illustrates in flow diagram form the possible transitions illustrated in table form in Fig. 4. Fig. 5 is a state transition table illustrating allowable sequences of simple characters in the Hindi language and includes allowable character replacement sequences. Fig. 5A is a state transition diagram illustrating allowable sequences of simple characters in the Hindi language. Fig. 5A illustrates in flow diagram form the possible transitions illustrated

in table form in Fig. 5. Fig. 6 illustrates a state transition table illustrating allowable sequences of simple characters in the Vietnamese language and includes allowable character replacement sequences. Fig. 6A illustrates a state transition diagram illustrating allowable sequences of simple characters in the Vietnamese language. The state transition diagram of Fig. 6A illustrates in flow diagram form the possible transitions illustrated in table form in the state transition table illustrated in Fig. 6.

The tables and diagrams illustrated in Figs. 4 – 6A are based on the syntactical rules of the selected languages and are presented for illustrative purposes only. The syntactical rules for each represented language, including placement of simple characters relative to each other in formation of complex characters is well known to those skilled in the mechanics of those languages. Also, it should be well understood that the present invention is not limited to the illustrative languages used in this description, but is applicable to the formation of any rules-dictated sequence of characters of any language.

Referring to Fig. 4, the user types a specific character, such as a leading vowel (LV1 – LV4) in the Thai language, the sequence checking feature 215 of the present invention starts from an initial state of zero, and identifies the class or category of that character (in this case, a leading vowel). That class specifies the column in the state transition table or the starting point in the state transition flow diagram to look into. The sequence checking feature 215 then finds the transition state number and the predefined series of state transition actions to apply to the existing sequence of characters (in this case, a sequence of one character—the leading vowel).

The transition state numbers indicate the next state to start from before treating the next input character. If the next state is zero, then the sequence checking feature 215 has reached a valid and complete sequence. If the next input character does not transition to any other state number then the character is not valid and is not combined with or

appended to the existing sequence. Once a determination is made that the typed character is not valid, the sequence checking feature 215 prevents the input of the typed simple character, and the user is forced to type an alternate simple character.

For example, referring still to Fig. 4, starting with the input of a leading vowel (LV1), the state transition table allows two possible transition states, namely states 40 and 9. That is, after the user types in a next character after the leading vowel, that next character is located in the row at the top of the table. Then, the column under the newly-typed character is followed to find possible transition states. Cells that are shaded gray contain transitions for use by the character type and replace feature 220 and are discussed below. If the type and replace feature 220 is turned off, the cells shaded in gray are not used. A blank cell in the state transition table indicates that the newly input character may not be used according to the rules of the selected language. The additional information contained in each field also is described below.

Referring now down the "state" column, it is seen that state 40 represents a character sequence context of LV1LV1 which means that if the user types another leading vowel after the initial leading vowel (i.e., a trailing vowel since it will follow the first leading vowel), the second vowel will be allowable under the rules for the Thai language. State 9 shows that the addition of a consonant (C) is an allowable next character.

However, if for example, the user types a tone mark (T) after typing the first leading vowel, that simple character will not be displayed because a leading vowel followed by a tone mark is not allowed and is not one of the transition states pointed to in the state transition table after the input of the leading vowel. Referring to Fig. 4A and keeping with the present example, starting with any leading vowel (LV1 – LV4) it is seen that the only allowable flow is to another vowel or a consonant.

It should be understood that for some languages, only a certain set of simple characters may be typed first. That is, if other than an allowable first character is typed, the character will not be displayed. Referring to Figs. 4 and 4A, for the Thai language, it is seen that the only characters that may be typed first include non-composible (NON) Thai characters, such as English language punctuation marks, control keys (CTRL), spaces, long tail consonants (LT), leading vowels (LV) and consonants (C). The tables and flow diagrams illustrated in Figs. 5, 5A, 6 and 6A are used and operate in a manner like that described for the table and flow diagram illustrated in Figs. 4 and 4A.

The sequence checking feature 215 does not handle replacement or deletion of existing simple characters during the input of a sequence of simple characters. As described above, that feature prohibits the addition of an invalid simple character onto an otherwise valid sequence of simple characters. In accordance with an exemplary embodiment of the present invention, the character type and replace feature 220 complements the sequence checking feature 215.

The character type and replace feature 220 allows for the automatic replacement of parts of the sequence of simple characters being validated by the sequence checking feature 215 without having to move the cursor back to a position at which the edit needs to take place. Fig. 12 illustrates input of simple Hindi characters acted upon by the functionality of the character type and replace feature 220. As illustrated in Fig. 12, the last simple character input by the user may not be appended to the sequence of previously input characters without violating the rules of the Hindi language. Therefore, the character type and replace feature 220 removes the first character of the sequence and inserts the new character (last character input) and inserts the new character before the last character of the sequence. Operation of type and replace actions used by the character type and replace feature 220 is described in detail below.

The character type and replace feature 220 is optional and is triggered when the sequence checking feature 215 invalidates an input character. As described above, the character type and replace feature 220 must be turned on by the user if it is not set on by default. Because the character type and replace feature 220 is complementary to the sequence checking feature 215, the sequence checking feature 215 must be on in order to use the character type and replace feature 220.

When the sequence checking feature 215 invalidates an input character, the character type and replace feature 220 tries to replace and/or combine an existing simple character in the validated sequence with the newly input simple character according to the rules for the selected language. If the character type and replace feature 220 succeeds, the new complex character is displayed. If not, the complex character is not changed.

The character type and replace feature 220 is implemented as a state transition machine similar to the implementation of the sequence checking feature 215 described above. As with the sequence checking feature 215, implementation of the character type and replace feature 220 is based on a table of transitions. At typing time, a specific language, such as Thai language, is selected by the user. This language will specify to the character type and replace feature 220 which table of transition to use. The state transition table implemented by the character type and replace feature 220 provides the character type and replace feature 220 with character replacement actions based on the rules for the selected language.

Referring to Figs 4, 5 and 6, the cells in the transition tables shaded in gray represent character type and replace actions which may be taken by the type and replace feature 220 to insert a newly typed character into a previously validated sequence or to replace a previously accepted simple character with the newly typed simple character. For example, referring to Fig. 5, say the user has selected the Hindi language for text

input and has initially typed an independent vowel (Iv) followed by an accent (Ac). Next the user types a vowel sign (Vs). First, the state of the current sequence context (IvAc) is determined by looking down the "context" column to find state 6 corresponding to context (IvAc). Next, the row containing state 6 is followed to the right to the cell under the newly typed character (Vs) located in the "character" row at the top of the table. That cell which is shaded in gray indicates the new character may not be appended to the current sequence and the cell contains the type and replace action "8i1" for use by the type and replace feature 220 in dealing with the newly typed character.

Fig. 9 is a table of character type and replace actions. Fig. 9A is a table of character type and replace actions for the Thai language only. Referring to Fig. 9, the action "8i1" is translated according to the symbol "Char i[n]," where i = 1, to direct that the new character (Vs) should be inserted into the current sequence one "1" character from the right or just to the left of the accent (Ac). The "8" in the action indicates the state of the sequence to transition to after the type and replace action is accomplished. Referring back to Fig. 5, and looking down the "state" column to state "8," it is seen that the corresponding sequence context is (IvVsAc). It can be seen for the example sequence of (IvVsAc) that the next state for this sequence, found in the column to the right of the "state column," is zero (0). Accordingly, the sequence is a complete and valid sequence. Therefore, if a new character is typed by the user that new character will begin a new sequence. The character type and replace functionality works only on a valid sequence of characters. That is, the character type and replace functionality modifies a valid sequence of characters with an invalid character to generate another valid sequence of characters.

Accordingly, while the rules of the Hindi language did not allow the vowel sign (Vs) to be appended to the sequence of the independent vowel and the accent, the replace action taken by the type and replace feature 220, inserted the vowel sign into the sequence in a manner consistent

with the rules of the selected language (in this case Hindi). Referring back to Fig. 9, it can be seen that other actions include, for example, "a" for append the new character, "d1" for remove the previous character, "dn" for remove the nth character, "r1" for replace the previous character with the new character, and "rn" for replace the nth character with the new character.

It is advantageous to describe the operation of the exemplary features of the present invention by way of an exemplary flow diagram. Fig. 7 is a flow diagram illustrating the preferred steps for character sequence checking and type and replace for a newly input character. At step 700, the method begins and moves to step 710 where the transition state for sequence checking is set to an initial state of zero. At step 715 a determination is made whether a new character has been input. If not the method ends at step 705. If a new character has been input, the method proceeds to step 720 and gets the new character for sequence checking in accordance with the rules of the selected language.

At step 725, a determination is made whether the new character is a member of the set of characters comprising the selected language. If the new character is not a character of the selected language, the method proceeds to step 730 and the new character is appended to the sequence. The method then proceeds to step 710 where the transition state is reset to zero so that the next input character will begin a new sequence. As discussed above, when a non-selected language character is input such as an English language punctuation mark or space, that character is allowed, but the selected language character will start a new sequence.

If at step 725 the newly input character does belong to the set of characters comprising the selected language, the method proceeds to step 735. At step 735, a determination is made whether there is a state transition from the current state for the newly input character. As discussed above, there will be a state transition from the current state if the cell in the state transition table (Figs. 4, 5, 6) corresponding to the new character contains

an action to append the new character. No transition from the current state exists if the cell is blank or if the cell contains only type and replace actions.

If at step 735 there is a state transition from the current state, that is, the new character may be appended to the last character or the new character may be initially input to start a sequence, then the method follows the "yes" branch and proceeds to step 740. At step 740, the new character is accepted as the initial character or is appended to the last character. For example, referring to Fig. 5, if the new character input is an independent vowel (Iv) in the Hindi language, then the transition state in the cell corresponding to the new character includes the action "5a" which indicates the character may be appended and the new state of the sequence is state 5. State 5 is accordingly the starting state for the next input character.

The method then proceeds to step 760 where the state is updated. For the present example, the state is updated to state 5. The method then proceeds to step 715 to determine whether there is a new character. If there is no new character, the method ends at step 705.

If at step 735, there is no state transition from the current state, as in the case of an invalid character, the method follows the "no" branch to step 745. At step 745, a determination is made whether the functionality of the character type and replace feature 220 is turned on. If the character type and replace feature 220 is not turned on, the method follows the "no" branch and proceeds to step 770 where the newly typed character is discarded and is prohibited from display. Continuing with the example set forth above, say the next character typed after the independent vowel (Iv) is a dependent vowel (Dv) in the Hindi language. Referring to Fig. 5, following down the column under the Dv character to state 5 (previously set) yields a blank cell which indicates no transition from the last state. Accordingly, the new character may not be appended to the sequence comprised of the independent vowel (Iv) and so the user must input a new character at step 715. If the user does not input a new character then the method ends at step 705.

If at step 745 the character type and replace feature 220 is turned on, the method follows the "yes" branch and proceeds to step 750. At step 750 a determination is made as to whether there is a type and replace action on this state. In the present example, the user typed in a dependent vowel after an independent vowel in the Hindi language. As discussed above, referring to the state transition table in Fig. 5, the input of the dependent vowel (Dv) after the independent vowel (Iv) yielded a blank cell which indicates there is no transition state from the previous state. That blank cell also indicates that there are no available type and replace actions for this character. In other words, the previously validated character (Iv) cannot be replaced with the new character (Dv) to yield a valid sequence of one character beginning with the dependent vowel because dependent vowels may not come first in a valid Hindi character sequence.

Say, for example, that the current sequence at step 750 is an independent vowel followed by an accent (IvAc) in the Hindi language and the new character typed by the user is a vowel sign (Vs). This example was fully discussed above. At step 750, a determination is made from the state transition table in Fig. 5 that the type and replace action for this character is "8i1." The method proceeds to step 755 where the type and replace action is performed. As discussed above, the action "8i1" directs the newly typed character to be inserted into the sequence before the first character on the right. In the present example, the vowel sign (Vs) is inserted before the accent (Ac) to yield the sequence of an independent vowel followed by a vowel sign followed by an accent (IvVsAc).

The method then proceeds to step 760 and updates the transition state. In the present example, the transition state is updated to state 8 which is the state of the three-character sequence (IvVsAc). The method then proceeds to step 715 to determine whether there is a new character. If there is no new character, the method ends at step 705. Referring to Fig. 5, it can be seen for the example sequence of (IvVsAc) that the next state for this sequence (found in the column to the right of the

"state column," is zero (0). Accordingly, the sequence is a complete and valid sequence. Therefore, if a new character is typed by the user at step 720, that new character will begin a new sequence.

The sequence checking feature 215 is applied during the sequential keyboard entry of characters. If the user moves the cursor which changes the sequence being validated, the sequence checking feature 215 reconstructs the sequence context before restarting the validation with newly entered characters. While typing a complex character sequence, the user may leave the sequence by using Backspace, Delete, typing a non-selected language character, or by moving the cursor somewhere else in the text to begin typing complex characters again. In this case, the sequence context reconstruction feature 225 will reconstruct the context of the complex character in order to resynchronize the sequence checking feature 215 with the right state in the state transition table. If the user opens a document which was typed without the sequence checking feature 215, the sequence context reconstruction feature 225 may have to undo what the user typed in order to reconstruct the sequence context of a given sequence.

Fig. 8 is a flow diagram illustrating the preferred steps for reconstructing the context (i.e., sequence of characters) of a previously input sequence of characters. In general the context or character sequence of a previously input sequence of characters is determined by placing the cursor of the user's computer program at some location to the right of a sequence of simple characters or complex character and identifying the first complete complex character to the left of the cursor. This is accomplished by identifying the simple character that starts the complex character from the very last simple character of the sequence (which is the character at the left of the cursor). The identification process goes from right to left starting with the character at the left of the cursor and ending on the first character of the identified complex character or ending on an invalid character after a specified number of character scans according to the selected language.

The method begins at step 800 and moves to step 810. At step 810 the maximum number "n" of characters which may comprise a complex character in the selected language is established. For example, say the selected language is Hindi, and in Hindi the maximum number of simple characters which may comprise a Hindi complex character is five (5). The method then moves to step 820 where the cursor is placed at some location in the text typed in the selected language.

In the present example, say the cursor is placed to the right of a complex character typed in Hindi and comprised of an independent vowel followed by a vowel sign followed by an accent (IvVsAc). The method then proceeds to step 825 where a determination is made whether the last character (Ac) is the beginning of a sequence which would yield a next state of zero (0). That is, if the next state after entry of the character (Ac) is zero, then that character is a complete sequence and the sequence context has been reconstructed which means the sequence checking feature 215 may begin checking the next sequence input by the user.

Because the last character did not yield a next state of zero, the method follows the "no" branch and proceeds to step 830 where the method moves to the next character and then proceeds back to step 825. At step 825, a determination is made whether the last two characters complete a sequence which yields a next state of zero. In the present case the last two characters (VsAc) do not yield a next state of zero. Accordingly, the method follows the "no" branch to step 830 and moves back to the third character from the right before proceeding back to step 825.

At step 825, a determination is made as to whether the sequence of the three characters (IvVsAc) complete a sequence which yields a next state of zero. With reference to Fig. 5, it can be seen that this sequence does yield a next state of zero which means the sequence is a valid and complete sequence in accordance with the rules of the selected language. The method then proceeds to step 835 and finds the final state of the three-character sequence or state 8 for the present example. This state is used by

the sequence checking feature 215 to determine where to start from in checking the next sequence of characters.

If the method goes back at step 830 until it reaches the maximum number of characters that may comprise a complex character under the selected language without finding a complete and valid sequence, then that sequence of characters includes one or more invalid characters. This would be the case where the sequence was input with the sequence checking feature turned off. In this case the user can simply delete the sequence and re-type the sequence.

As described herein, the present invention provides for a method and system for checking the validity of a sequence of input characters according to the syntactical rules of a selected language. If an input character may not be appended to the previously input sequence according to the rules of the selected language, the newly input character may be prohibited from being appended to the sequence and displayed on the user's computer. The present invention provides for editing of previously input character sequences by determining the validity context of sequences of characters previously input.

While this invention has been described in detail with particular reference to exemplary embodiments thereof, it will be understood that variations and modifications can be effected within the spirit and scope of the invention as described herein and as defined in the appended claims.

CLAIMS

What is claimed is:

1. A method of checking a sequence of input characters according to rules of a selected language, comprising the steps of:

receiving a first character;

determining whether the first character may begin a valid sequence of characters according to rules associated with the selected language;

if the first character may begin a valid sequence of characters according to rules associated with the selected language, accepting the first character for display; and

if the first character may not begin a valid sequence of characters according to rules associated with the selected language, prohibiting accepting the first character for display.

2. The method of Claim 1, further comprising the steps of:

receiving a second character;

determining whether the second character may be appended to the first character according to rules associated with the selected language;

if the second character may be appended to the first character according to the rules associated with the selected language, appending the second character sequentially to the first character; and

if the second character may not be appended to the first character according to the rules associated with the selected language, prohibiting appending the second character to the first character.

3. The method of Claim 1, wherein the step of determining whether the second character may be appended to the first character according to rules associated with the selected language includes the steps of:

in a state transition table, assigning a first state to the first character according to the rules associated with the selected language;

assigning a second state to the second character according to the rules associated with the selected language;

determining whether the state transition table includes a state transition from the first state to the second state; and

if the state transition table includes a state transition from the first state to the second state, determining the second character may be appended to the first character according to the rules associated with the selected language; and

if the state transition table does not include a state transition from the first state to the second state, determining the second character may not be appended to the first character according to the rules associated with the selected language.

4. The method of Claim 3, further comprising the steps of:

determining whether appending the second character to the first character creates a complete sequence of characters according to the rules associated with the selected language;

if the sequence of characters is a complete sequence of characters according to the rules associated with the selected language, determining whether a third input character may begin a second valid sequence of characters according to rules associated with the selected language;

if the third character may begin a second valid sequence of characters according to rules associated with the selected language, accepting the third character for display; and

if the third character may not begin a second valid sequence of characters according to rules associated with the selected language, prohibiting accepting the third character for display.

5. The method of Claim 4, wherein the step of determining whether appending the second character to the first character creates a complete sequence of characters according to the rules associated with the selected language includes the step of:

determining whether the second state points to a third transition state representing a reset transition action.

6. The method of Claim 1, wherein the step of appending the second character to the first character includes the step of combining the first and second characters according to the rules associated with the selected language to form a single complex character.

7. The method of Claim 1,

wherein the step of prohibiting appending the second character to the first character includes the step of prohibiting the display of the second character with the first character on a display screen, and

wherein the step of appending the second character to the first character includes the step of displaying the second character with the first character on the display screen.

8. The method of Claim 1, wherein the first character is a simple character from the selected language.

9. The method of Claim 1, wherein the first character is comprised of two or more simple characters from the selected language.

10. The method of Claim 1, wherein the second character is a simple character from the selected language.

11. The method of Claim 1, wherein the selected language is Thai.

12. The method of Claim 1, wherein the selected language is Hindi.

13. The method of Claim 1, wherein the selected language is Vietnamese.

14. A computer-readable medium on which is stored a computer program for checking a sequence of input characters according to rules of a selected language, the computer program comprising instructions, which when executed by a computer, perform the steps of:

receiving a character;

determining whether the character may be appended to a previous character to form a sequence of characters according to rules associated with the selected language;

if the character may be appended to the previous character according to the rules associated with the selected language, appending the character to the previous character to form a sequence or characters according to the rules associated with the selected language; and

if the character may not be appended to the previous character according to the rules associated with the selected language, prohibiting appending the character to the previous character.

15. The method of Claim 14, further comprising the steps of:

determining whether the sequence of characters is a complete sequence in accordance with the rules associated with the selected language;

if the sequence of characters is a complete sequence of characters according to the rules associated with the selected language prohibiting appending additional characters to the sequence of characters.

16. A method of checking a sequence of input characters according to rules of a selected language, comprising the steps of:

receiving an input character;

if the character is not associated with the selected language, displaying the character;

if the character is associated with the selected language, determining whether the character may be displayed as a single character according to the rules of the selected language;

if the character may not be displayed as a single character according to the rules of the selected language, determining whether the character may be appended to one or more additional characters to form a valid sequence of characters according to the rules of the selected language;

if the character may not be appended to one or more additional characters to form a valid sequence of characters, discarding the character; and

if the character may be appended to one or more additional characters to form a valid sequence of characters, displaying the character.

17. A method of establishing the sequence validation context of a sequence of characters comprising a complex character, comprising the steps of:

determining the maximum number of characters that may comprise a valid sequence of characters according to the rules of a selected language;

beginning with the last simple character of a sequence of characters, determining whether the last character is valid as a complete sequence of characters comprising a complex character,

if the last character of the sequence of characters is valid as a complete sequence of characters comprising a complex character, then returning the context of the input character as a context for a complex character;

if the input character of a sequence of characters is not valid as a complete sequence of characters comprising a complex character, then determining whether a combination of the last character and the character input immediately to the left of the last character is valid as a complete sequence of characters comprising a complex character,

if the combination of the last character and the character input immediately to the left of the last character is valid as a complete sequence of characters comprising a complex character, then returning a context for the combination as the context for a complex character;

if the combination is not valid as a complete sequence of characters comprising the complex character, then determining whether the combination combined with the next character to the left of the combination is valid as a complete sequence of characters comprising a complex character, and if not, then creating subsequent combinations of characters by adding one at a time additional characters input to the left of the last subsequent combination until the maximum number of characters that may comprise a valid sequence have been combined to form a sequence of characters that may be checked for validity as a complete sequence of characters comprising a complex character; and

if any said subsequent combination of characters are valid as a complete sequence of characters comprising a complex character according

to the rules of the selected language, then returning a context for said any said subsequent combination as the context for a complex character.

18. A system for checking a sequence of input characters according to rules of a selected language, comprising:

a computer program module operative

to receive a first character;

to determine whether the first character may be the first character of a sequence of characters according to the rules associated with the selected language;

to receive a second character;

to determine whether the second character may be appended sequentially to the first character according to rules associated with the selected language;

to append the second character sequentially to the first character if the second character may be appended to the first character according to the rules associated with the selected language; and

to prohibit appending the second character to the first character if the second character may not be appended to the first character according to the rules associated with the selected language.

"METHOD AND SYSTEM FOR CHARACTER SEQUENCE CHECKING ACCORDING TO A SELECTED LANGUAGE"

ABSTRACT OF THE DISCLOSURE

A method and system for checking the validity of a sequence of input characters according to the syntactical rules of a selected language is provided. If an input character may not begin a valid sequence of characters, or if the input character may not be appended to a previously input sequence of characters according to the rules of the selected language, the newly input character may be prohibited from being displayed singularly or appended to the previous sequence and displayed on the user's computer. Previously input character sequences may be edited by determining the sequence validity context of previously input sequences of characters.

Attorneys' Docket No. 13237-2425

MS Docket No. 131071.1

6502.90 Searched

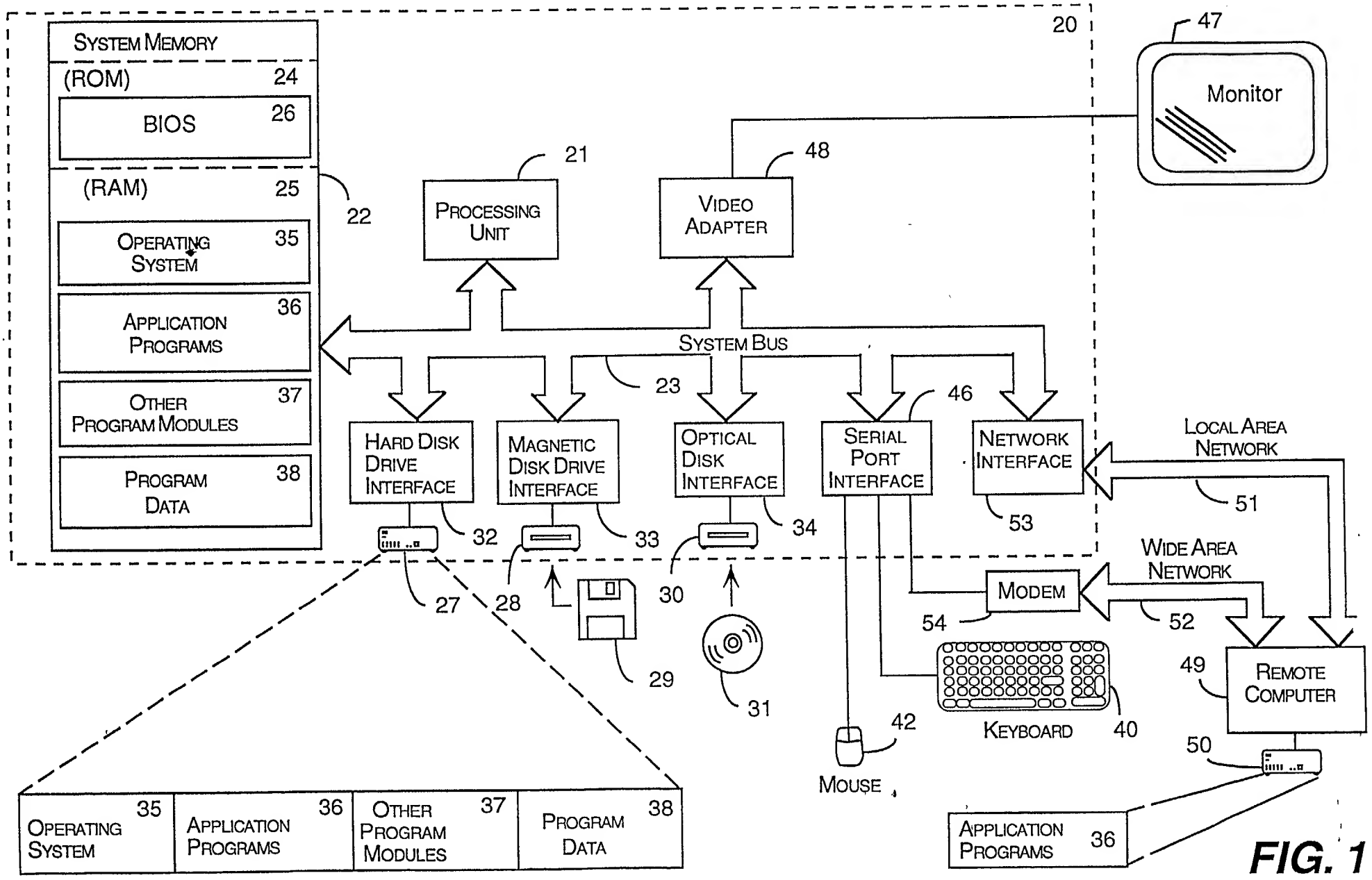

FIG. 1

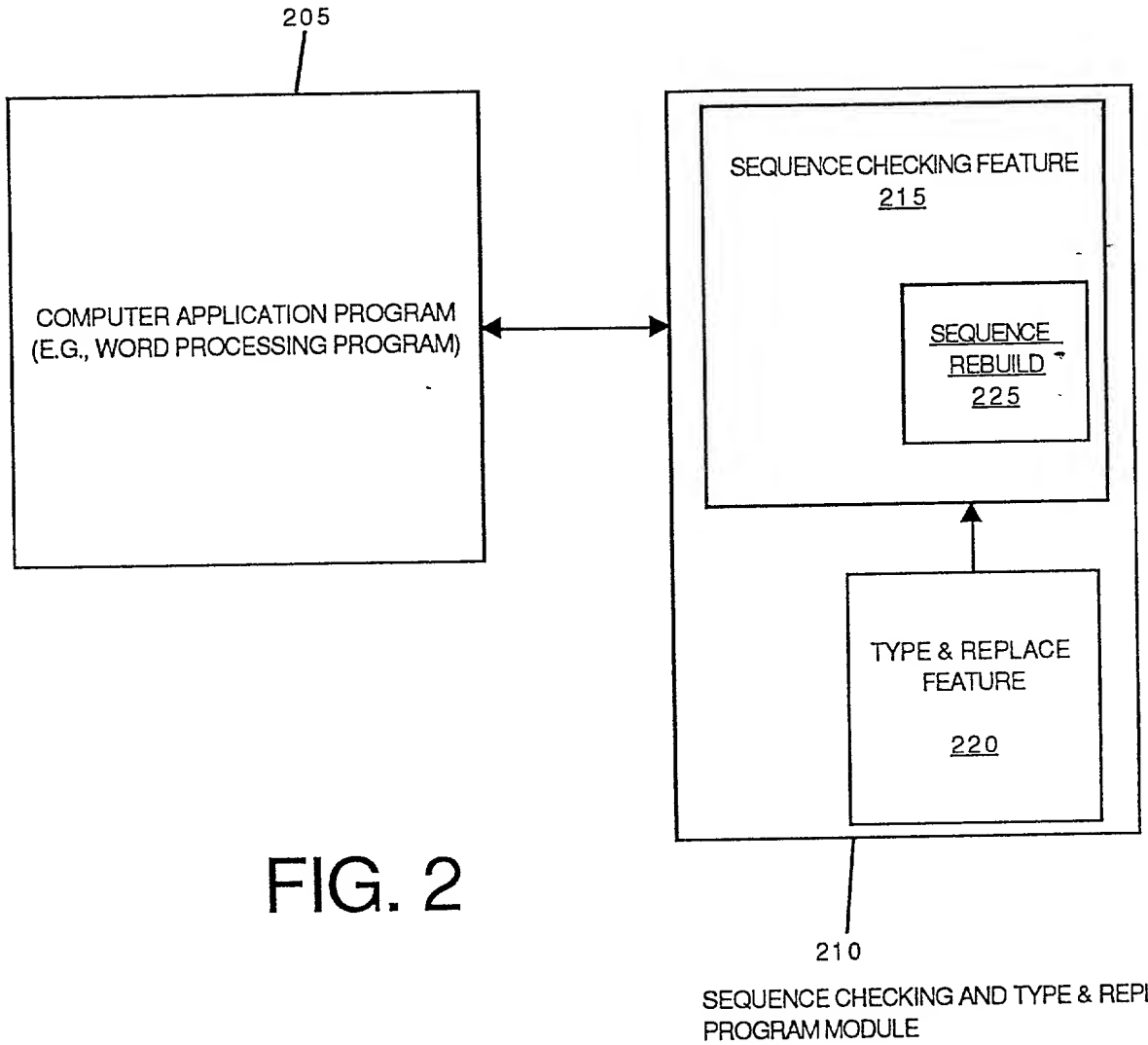
205
COMPUTER APPLICATION PROGRAM
(E.G., WORD PROCESSING PROGRAM)
SEQUENCE CHECKING FEATURE
215
SEQUENCE REBUILD
225
TYPE & REPLACE FEATURE
220
210
SEQUENCE CHECKING AND TYPE & REPLACE PROGRAM MODULE

FIG. 2

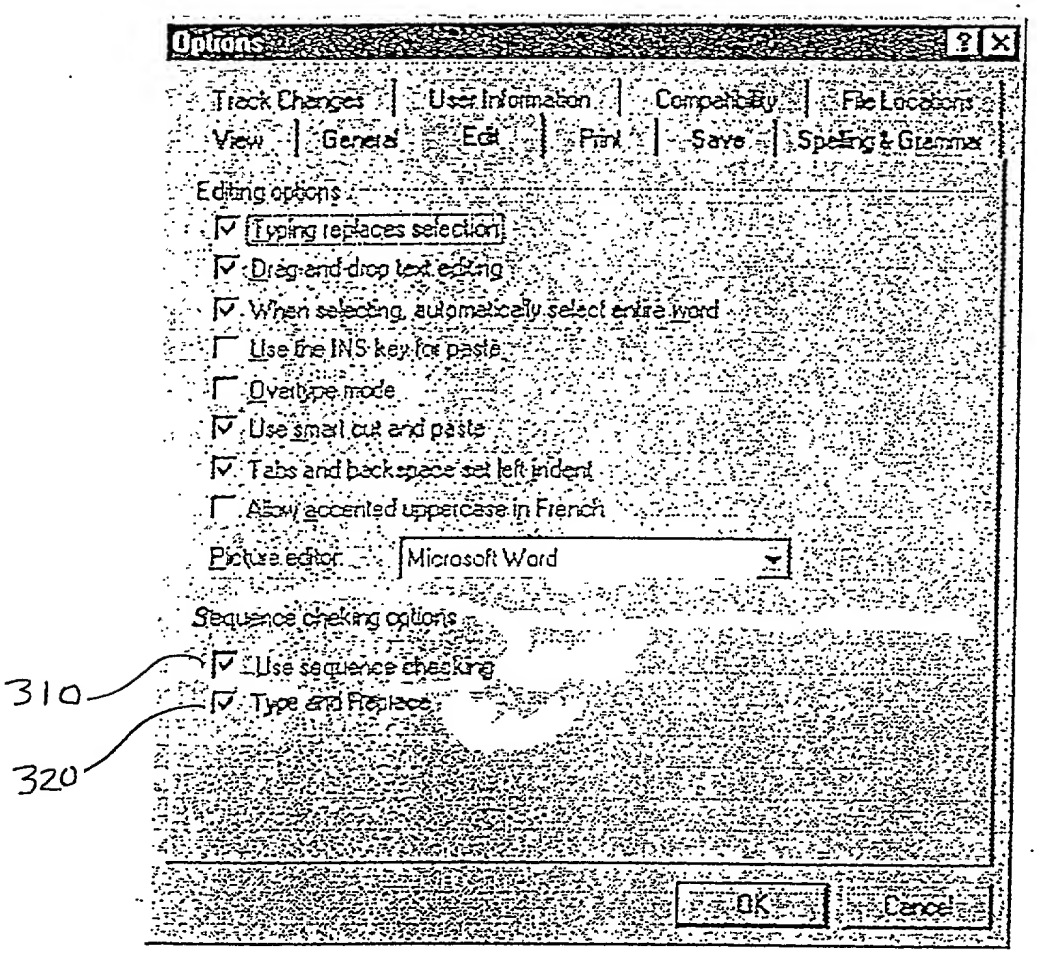
Options
Track Changes
User Information
Compatibility
File Locations
View
General
Edit
Print
Save
Spelling & Grammar
Editing options
Typing replaces selection
Drag-and-drop text editing
When selecting, automatically select entire word
Use the INS key for paste
Overtype mode
Use smart cut and paste
Tabs and backspace set left indent
Allow accented uppercase in French
Picture editor:
Microsoft Word
Sequence checking options
Use sequence checking
Type and Replace
310
320
OK
Cancel

Fig. 3

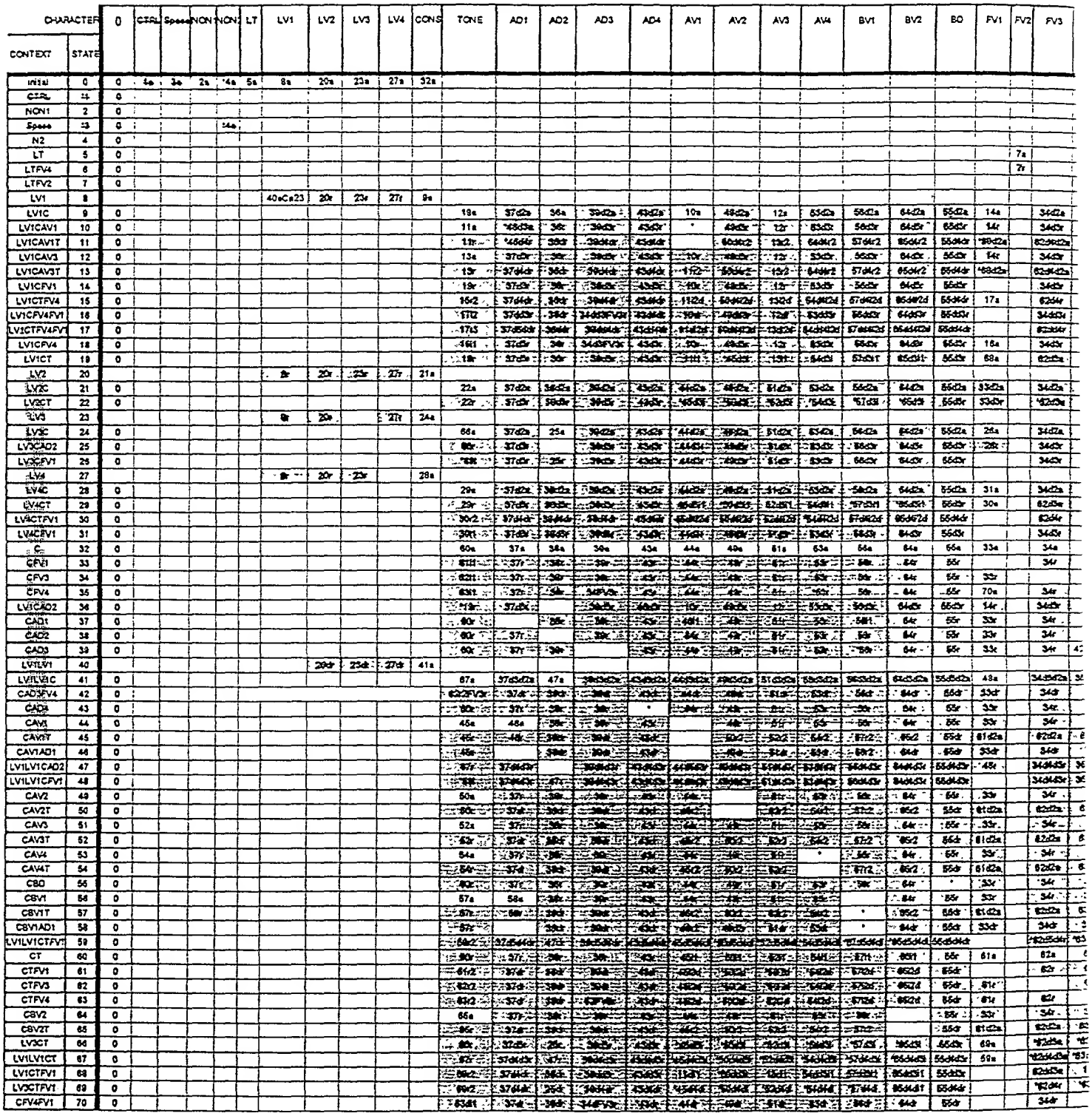

Fig. 4

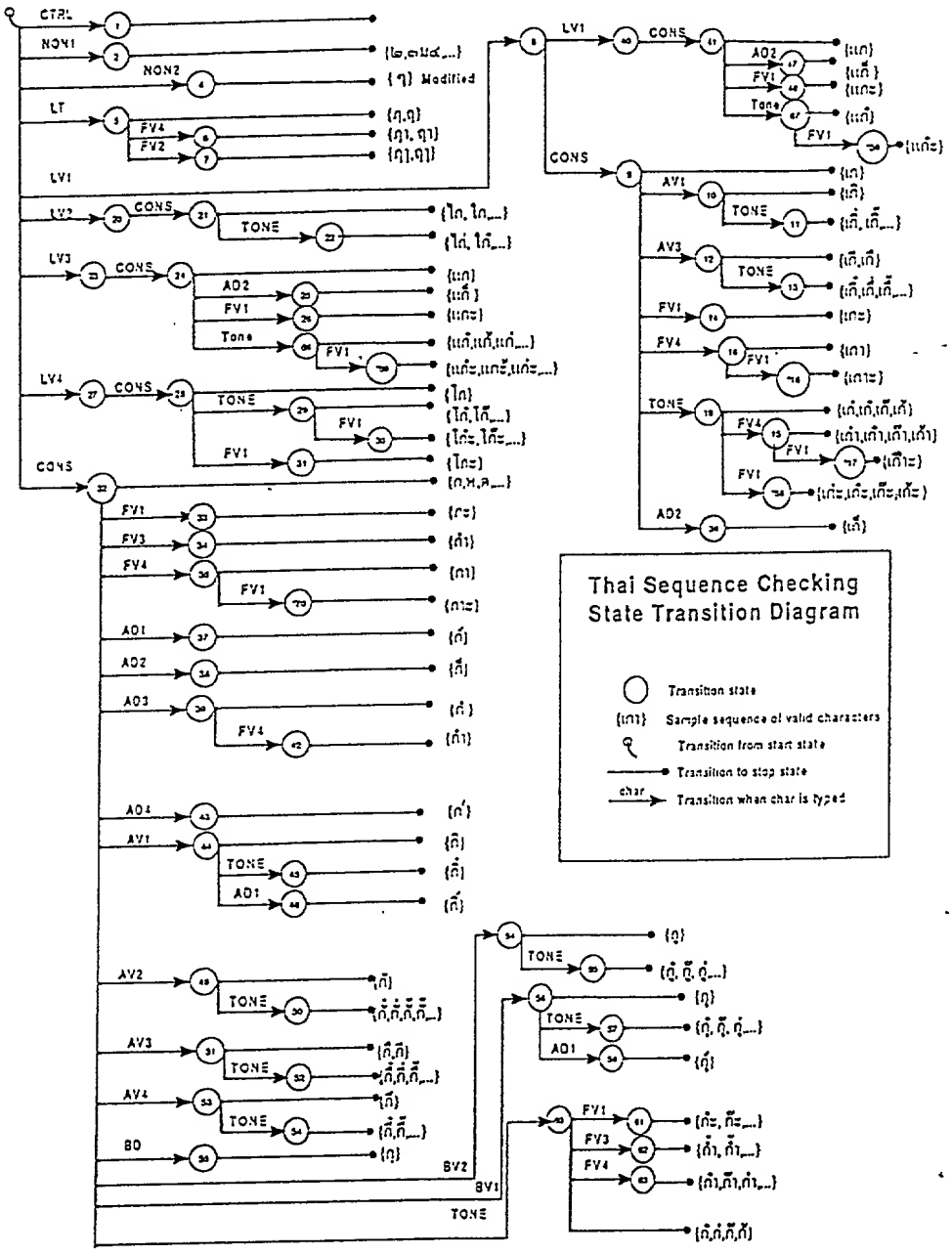
Thai Sequence Checking
State Transition Diagram
Transition state
{ก1} Sample sequence of valid characters
Transition from start state
Transition to stop state
char Transition when char is typed
CTRL
NON1
NON2
LT
FV4
FV2
LV1
LV2
LV3
LV4
CONS
TONE
AD2
FV1
Tone
AV1
AV3
FV3
AD1
AV2
AV4
BD
BV2
BV1

Fig. 4A

CHARACTER		{}	ind_chars	abbr	delimit	number	ind_vowel	norm_conso	ext_conso	virama	nukta	dep_vowel	vowel_sign	accent	zerowidth chars
CONTEXT	STATE		Ic	Ab	D	Nb	Iv	Nc	Ec	V	Na	Dv	Vs	Ac	Zw
initial	0	0	1a	2a	3a	4a	5a	9a	10a						
Ic	1	0													
Ab	2	0													
D	3	0													
Nb	4	0													
Iv	5	0											7a	6a	
IvAc	6	0											8i1	6r1	
IvVs	7	0											7r1	8a	
IvVsAc	8	0											8r2	8r1	
Nc	9	10									10a				
NcNa	10	0								11a		14a	18a	20a	
NcNaV	11	12													12a
NcNaVZc	12	*						13a	10a						
NcNaVNc	13	10									10a				
NcNaDv	14	0										14r1	16a	15a	
NcNaDvAc	15	0										16r2	17i1	15r1	
NcNaDvVs	16	0										16r2	16r1	17a	
NcNaDvVsAc	17	0										17r3	17r2	17r1	
NcNaVs	18	0										16i1	18r1	19a	
NcNaVsAc	19	0										17i2	19r2	19r1	
NcNaAc	20	0										15i1	19i1	20r1	

Fig. 5

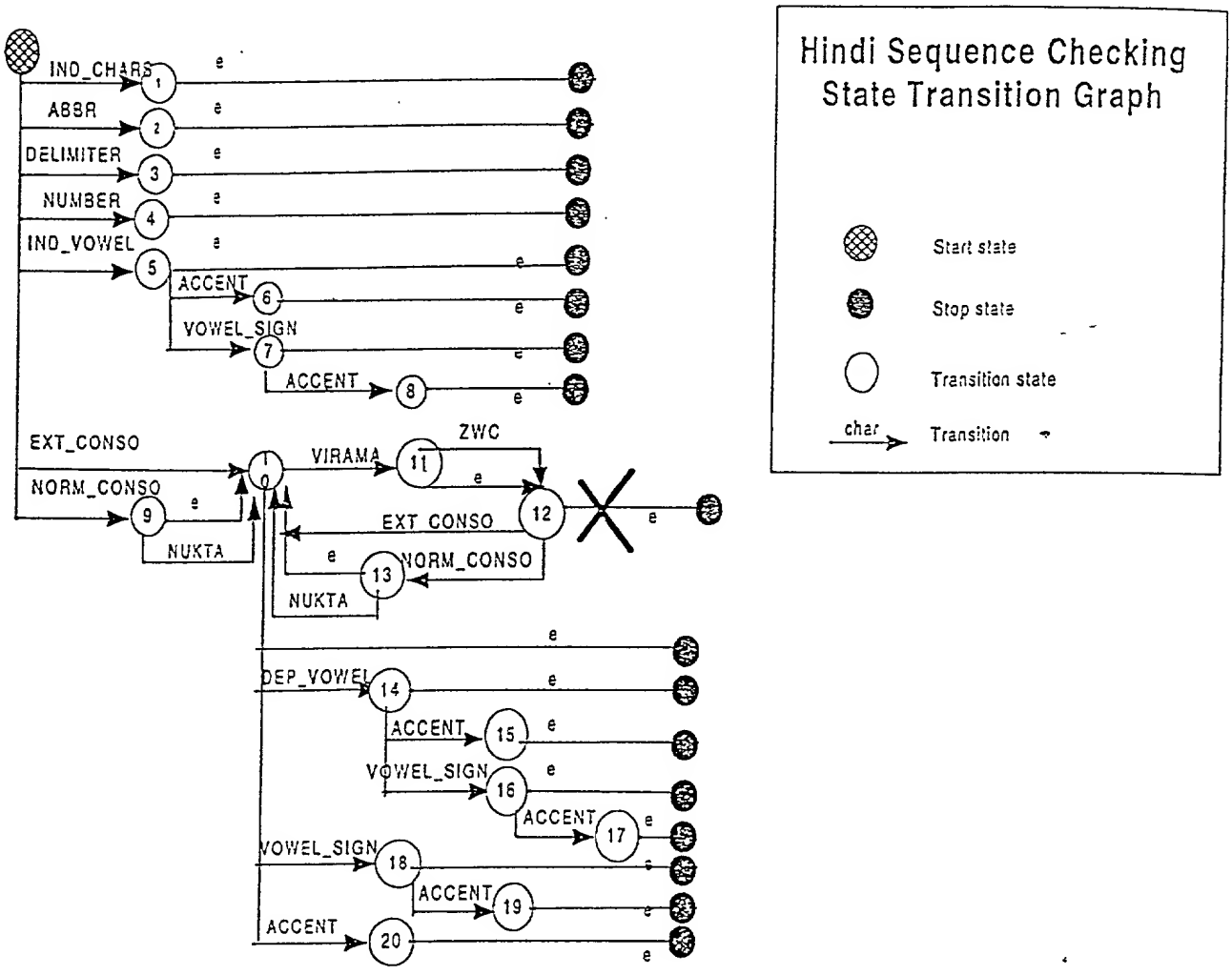
Hindi Sequence Checking
State Transition Graph
Start state
Stop state
Transition state
char
Transition
IND_CHARS
ABBR
DELIMITER
NUMBER
IND_VOWEL
ACCENT
VOWEL_SIGN
ACCENT
EXT_CONSO
NORM_CONSO
NUKTA
VIRAMA
ZWC
EXT CONSO
NORM_CONSO
NUKTA
DEP_VOWEL
ACCENT
VOWEL_SIGN
ACCENT
VOWEL_SIGN
ACCENT
ACCENT
e

Fig. 5A

VIETNAMESE SEQUENCE CHECKING
STATE TRANSITION GRAPH AND CHART

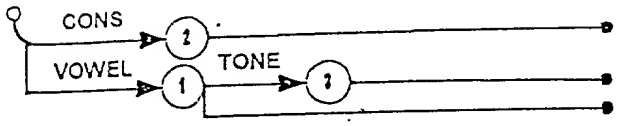

Fig. 6A

CHARACTER		{}	VOWEL	TONE	CONS
CONTEXT	STATE		V	T	C
initial	0	0	1a		2a
V	1	0		3a	
C	2	0			
VT	3	0		3r	

Fig. 6

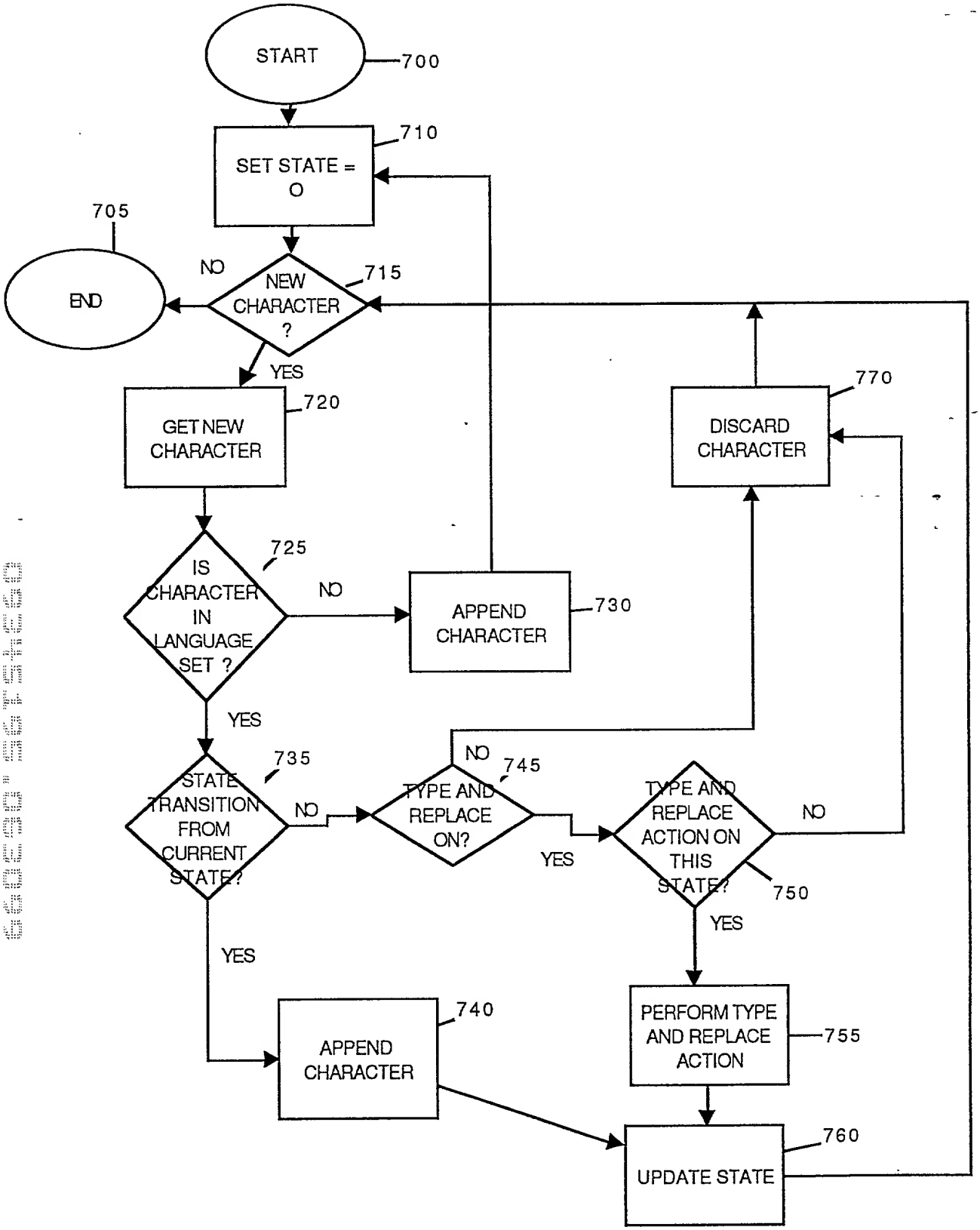
START
700
SET STATE = O
710
705
END
NO
NEW CHARACTER ?
715
YES
GET NEW CHARACTER
720
DISCARD CHARACTER
770
IS CHARACTER IN LANGUAGE SET ?
725
NO
APPEND CHARACTER
730
YES
STATE TRANSITION FROM CURRENT STATE?
735
NO
TYPE AND REPLACE ON?
745
NO
YES
TYPE AND REPLACE ACTION ON THIS STATE?
750
NO
YES
YES
APPEND CHARACTER
740
PERFORM TYPE AND REPLACE ACTION
755
UPDATE STATE
760

FIG. 7

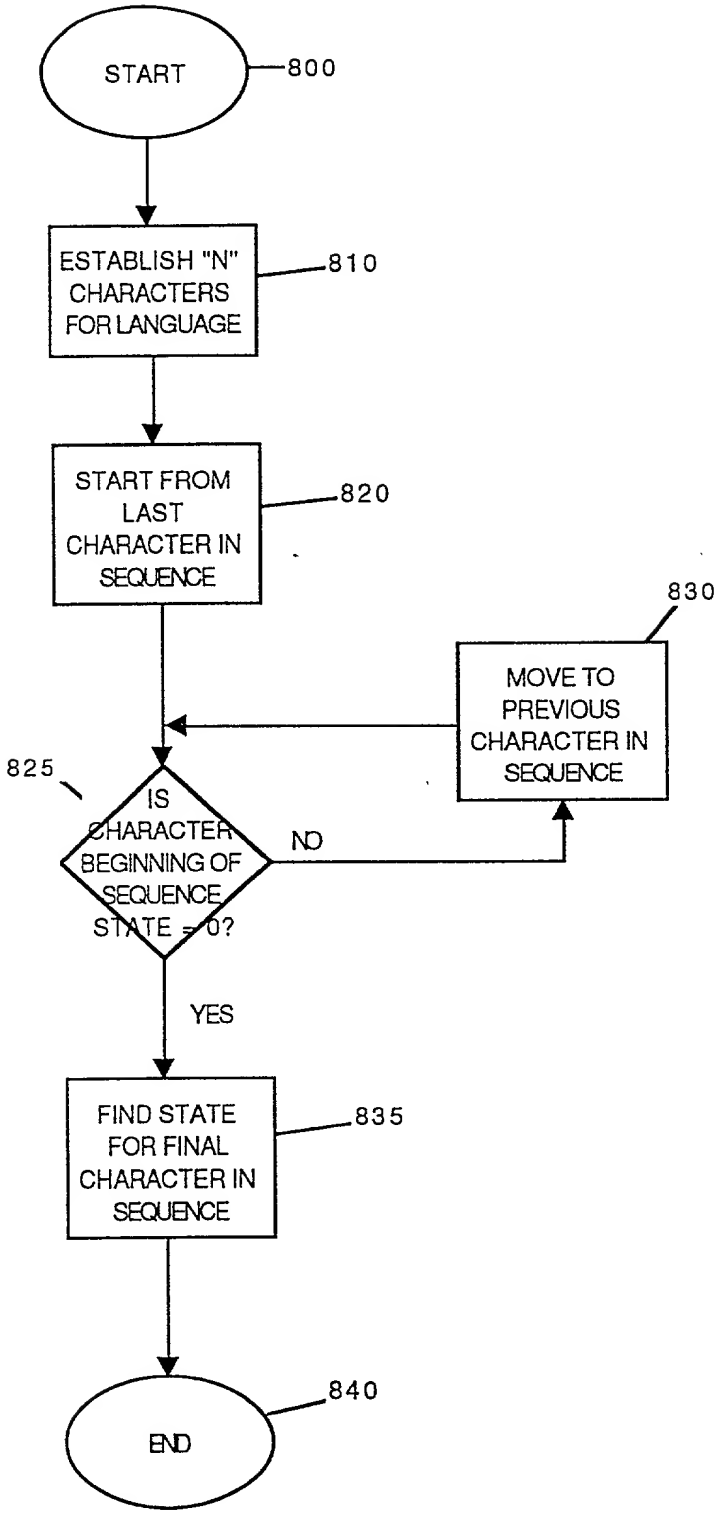
START
800
ESTABLISH "N" CHARACTERS FOR LANGUAGE
810
START FROM LAST CHARACTER IN SEQUENCE
820
830
MOVE TO PREVIOUS CHARACTER IN SEQUENCE
825
IS CHARACTER BEGINNING OF SEQUENCE STATE = 0?
NO
YES
FIND STATE FOR FINAL CHARACTER IN SEQUENCE
835
END
840

FIG. 8

SYMBOL	INDEX	ACTION	EXAMPLE
Char **a**	IP	Append the character char	AB ⇨ Ca → ABC
Char **I** [n]	IP–n where n>o	Insert the character Char at the index location (before the ith character from the right)	ABD ⇨ Ci] → ABCD
d[1]	IP–1	Remove the previous character	AB ⇨ d → A
dn	IP–n where n>1	Remove the character at the index location	ABC ⇨ d2 → AC
Char **r**[1]	IP–1	Replace the previous character with Char	AC ⇨ Br → AB
Char **rn**	IP–n where n>1	Replace the character Char at the index location with the new character	ADC ⇨ Br2 → ABC
Cxy		Contextual information (this does not necessarily generate an action) for a possible character composition by the client application. In the Thai transition table "**x**" has the following values: "**a**" for composing the last 2 characters into LV3 (Sara-AE) or "**b**" for composing the last 2 characters into FV3 (Sara-Am). "**y**" indicates the number of the next transition state.	ABC ⇨ Cx2 → AD (if x composes B and C into D)

Fig 9

SYMBOL	ACTION	EXAMPLE
FV3r	Substitute the last character with FV3	⇨ FV3r →

Fig 9A

keyboard input	Output
क	क
्	क्
क	क्क
ी	क्की

Fig 10

keyboard input	Output
क	क
्	क्
ी	क्

Fig. 11

keyboard input	Output
क	क
्	क्
य	क्य
ि	क्यि
ं	क्यिं
ो	क्यों

Fig. 12